Semantic Versioning is a way of versioning that has been recently getting more popular. (GeeksforGeeks, 2022a) It is also called SemVer (GeeksforGeeks, 2022a). It is a way to keep track of versions of software development projects, plugins, libraries, or other extensions that keep getting added throughout the life of the software. (GeeksforGeeks, 2022a)

The way Semantic Versioning works is the format of major.minor.patch. Major will increase on the breaking changes. Minor increases when new things are added that are backwards compatible, such as new HTTP Methods (Lauret, 2019b). And Patch is only increased when “modifications are made involve backwards compatible bug fixes.” (Lauret, 2019b) Also according to the book “Semantic Versioning applied to APIs consists of just two digits: BREAKING.NONBREAKING” (Lauret, 2019b) Versioning prior to 1.0 is part of the development phase, such as 0.8.2. Then once released as 1.0.0 this would be called version 1. (GeeksforGeeks, 2022a)

There are some advantages to Semantic Versioning. First its easy to track all the software transitions during each phase. It can also tell the developers what has changed at what time and what updates it may need. (GeeksforGeeks, 2022a) Next it assists in keeping “things clean and meaningful” (GeeksforGeeks, 2022a). Lastly it makes it easier for other developers who are going to use your software.

However, there are also some disadvantages of Semantic Versioning. The version starts at 0.1.0 during the development phase and not 0.0.1, considering that there have been no bug fixes and we are starting with some features. Also, Semantic Versioning “does not cover libraries tagged 0.\*.\*. The first stable version is 1.0.0.” (GeeksforGeeks, 2022a)

References:

GeeksforGeeks. (2022a). Introduction to Semantic Versioning. *GeeksforGeeks*. https://www.geeksforgeeks.org/introduction-semantic-versioning/

Lauret, A. (2019b). *The Design of Web APIs*. Manning.